# Optimizing Offer Sets in Sub-Linear Time

Vivek F. Farias
MIT Sloan School of Management, vivekf@mit.edu

Andrew A. Li
Tepper School of Business, aali1@cmu.edu

Deeksha Sinha
Operations Research Center, MITs, deeksha@mit.edu

Personalization and recommendations are now accepted as core competencies in just about every online setting, ranging from media platforms to e-commerce to social networks. While the challenge of estimating user preferences has garnered significant attention, the operational problem of using such preferences to construct personalized offer sets to users is largely still open, particularly in modern settings where a massive number of items and a millisecond response time requirement mean that even enumerating all of the items is impossible. Faced with such settings, existing techniques are either (a) entirely heuristic with no principled justification, or (b) theoretically sound, but simply too slow to work.

Thus motivated, we propose an algorithm for personalized offer set optimization that runs in time *sub-linear* in the number of items while enjoying a uniform performance guarantee. Our algorithm works for an extremely general class of problems and models of user choice that includes the mixture MNL model as a special case. We achieve a sub-linear runtime by leveraging the dimensionality reduction from learning an accurate latent factor model, along with existing sub-linear time approximate near neighbor algorithms. Our algorithm can be entirely data-driven, relying on samples of the user, where a 'sample' refers to the user interaction data typically collected by firms. We evaluate our approach on a massive content discovery dataset from Outbrain that includes millions of advertisements. Results show that our implementation indeed runs on the order of milliseconds, with increased performance relative to existing fast heuristics.

*History*:

## 1. Introduction

A common problem in modern web-services revolves around constructing personalized offer sets (or assortments) for users with the objective of optimizing some objective related to the user's experience with the service. Recommendation problems (such as those faced by services such as Netflix) represent a canonical version of such a problem. Assortment optimization problems (as faced by online retailers) are another common example. Rigorous latency constraints are an important practical consideration when building algorithms to construct optimized offer sets. A large-scale study recently showed that a 100ms delay in loading page content can result in a decrease in conversion of up to 7%[1]. When taken together with the vast size of the product universe (which can run to the tens of millions), such constraints place severe limitations on what a real-time algorithm to construct an optimized offer can do in practice. As such, practically implementable algorithms for real-time offer set optimization at scale must be sub-linear in the size of the universe of potential products.

The dynamic nature of the product universe limits the use of pre-computation. As will be evident later, distributional models of the user further restrict our ability to exhaustively pre-compute optimal offer sets. As a result, the dominant approach today to building optimized offer sets in real-time relies on the design of so-called approximate nearest neighbor (ANN) algorithms. Succinctly, such algorithms leverage metric-space representations of users and products wherein the distance between a user and product is inversely related to how attractive the product is to the user. Whereas such a metric space representation may be constructed offline, the real-time problem then consists of identifying a point in the metric space that corresponds to the user, and finding the $k$ products in the metric space that are nearest to that user in time sub-linear in the number of products. This problem is well-solved both theoretically and practically.

Now an economically grounded approach to constructing an optimal offer-set would typically rely on modeling the user's utility for various products. An offer set constructed assuming that a

[1] https://www.akamai.com/uk/en/about/news/press/2017-press/akamai-releases-spring-2017-state-of-online-retail-performance-report.jsp

user made choices to maximize utility would then seek not just to pick products that are likely interesting to the user but would further seek to account for substitution and complementarity effects. There is by now a vast literature dedicated to the estimation of models of choice as well as the associated assortment optimization problems given such models. The assortment optimization algorithms developed in this context are, while typically efficient, not sub-linear. On the other hand, the ANN paradigm while sub-linear is typically unable to account for assortment effects in a principled fashion which typically results in a slew of ad-hoc algorithmic tweaks. Here, we seek to begin bridging this gap.

**This Paper:** The present paper seeks to develop sub-linear time algorithms for offer-set optimization while allowing for rich, economically grounded models of customer choice. Specifically, like is typical in the ANN paradigm, we are endowed with a metric space. We are given a universe $V$ of $n$ products, wherein each product $v \in V$ is a point in the metric space. Similarly, a user $U$ is a random point in this space. Our objective is to solve, in sub-linear time, a problem of the form

$$\max_{S \subset V, |S| \leq k} \mathsf{E}\left[f(S,U)\right]$$

where the expectation is over $U$. The principle assumptions we place on the functions $f(\cdot,U)$ are that we require these functions be sub-modular and further that $f(\{v\},U)$ be non-increasing in the distance $d(U,v)$. As we discuss later this framework is quite flexible: for instance, it immediately captures the problem of picking an offer set to maximize conversion (i.e. the probability of a purchase) where consumer choice is driven by an essentially arbitrary random utility model.

*Our primary contribution is a sub-linear time algorithm to solve the optimization problem above with uniform performance guarantees.* We also present an empirical study on a large-scale corpus of page-view data (the dataset contains two billion page views of seven hundred million unique users) that establishes the value of our procedure.

Our key algorithmic contribution is a procedure to construct a specific sub-linear sized subset of products, in sub-linear time, via an approach we dub *locality-sensitive sampling.* Locality-sensitive sampling is a simple idea motivated by the same locality-sensitive hash functions that underly ANN algorithms. The set we construct enjoys the property that the optimal value of our optimization problem restricted to this set is close to the optimal value of the optimization problem over all products. As such, we then simply solve our optimization problem over this restricted set. A greedy algorithm trivially guarantees both a sub-linear run-time and a constant factor approximation.

The rest of this paper is organized as follows: we briefly review related work in the remainder of this section. Section 2 introduces our problem formally, along with our key modeling assumptions. Section 3 describes the motivation for our algorithm by way of an idealized sampling procedure. We then describe our actual algorithm, which is designed to approximate this idealized procedure, in Section 4. Finally, experimental results and conclusions given in Sections 5 and 6, respectively.

## 1.1. Related Work

There are at least three major streams of literature related to this work.

*Recommendation Algorithms* : In the area of recommender systems, the problem of learning user preferences from previous interactions has been studied extensively Jin et al. (2003, 2002), Freund et al. (2003), Schapire and Singer (1998). For the most part, successful learning algorithms work by embedding both users and items within some metric space such that a user's affinity toward an item is inversely related to their pairwise distance. See Adomavicius and Tuzhilin (2005) for an extensive survey of content-based, collaborative and hybrid recommendation approaches, and Zhang et al. (2019) for a survey of modern approaches based on deep learning.

A recent problem in this stream of literature is how to capture the impact of diversity in recommendations. These efforts have mostly focused on quantifying and maximizing diversity in recommendations sets. Kunaver and Požrl (2017) is an extensive survey of the research in this area. A key limitations of the current research here is that the diversity metric is not standardized. Moreover, increasing diversity has often been viewed as sacrificing accuracy of the recommendation set. We will take a more systematic approach to this.

*Assortment Optimization:* Another stream of literature related to our work is assortment optimization in the field of operations management. Assortment optimization is a principled modeling approach to choosing an optimal assortment to offer to customers.Kök et al. (2008) provides an overview of models found in literature and approaches common in practice. Integral to the assortment optimization problem is the model for user choice. One of the most well studied and commonly used choice model is the Multinomial Logit (MNL) model. The assortment optimization problem with the MNL choice model is tractable, even under various constraints (Talluri and Van Ryzin (2004), Rusmevichientong et al. (2010), Davis et al. (2013)). Though being attractive due to its tractability, the MNL models suffers from Independence of Irrelevant Alternatives (IIA) property. To overcome the IIA limitation, the Nested Logit (Williams (1977)) and Mixed Multinomial Logit (MMNL) models were proposed. More recently, assortment optimization has been studied under some new choice models like the Markov chain choice model (Désir et al. (2015)), distance-comparison based choice model (Kleinberg et al. (2017)), distribution over rankings (Farias et al. (2013)) and its variations (Désir et al. (2016)). One limitation of this stream of work is that sub-linear time algorithms effectively do not exist. Even linear time algorithms are rare and restricted to models like the simple multinomial logit that fail to capture user diversity.

*Approximate Nearest Neighbors* The third stream of literature that is relevant to our work is the problem of nearest neighbor (NN) search. In this problem, the goal is to pre-process the given data set so that the nearest neighbor to a query can be efficiently calculated. Chávez et al. (2001) give an overview of methods that have been proposed to solve this problem. Some sample works on the NN search problem are Omohundro (1989), Sproull (1991), Bentley (1975), and Yianilos (1993).

We focus on a particular type of approximate nearest neighbor search algorithm called Locality Sensitive Hashing (LSH) (Andoni and Indyk (2008)). Paulevé et al. (2010) describe various hash functions used in LSH algorithms. These have been used in several applications, but in particular find themselves used extensively in recommendation systems. This application has largely focused on obtaining binary representations of users and items which can then be used for doing fast similarity search computations Karatzoglou et al. (2010), Zhou and Zha (2012), Liu et al. (2014), Das et al. (2007), Liu et al. (2018), Zhang et al. (2014).

## 2. Model and Assumptions

We begin by introducing the core optimization problem that will be the subject of the rest of this paper. The problem is to select a personalized offer set that maximizes expected reward, subject to a cardinality constraint. Let $V$ denote the universe of items or products we can offer, and $k \in \mathbb{N}$ the maximum cardinality allowed, meaning the set of feasible offer sets is $\{S \subset V : |S| \leq k\}$. The need for personalization is driven by the notion of a user 'type': we assume that each user has a type, which takes values in some set $\mathcal{M}$, and that this type governs the reward we obtain for offering a given offer set. That is, the reward function, which we denote $f(\cdot,\cdot)$, is a map from $2^V \times \mathcal{M}$ to $[0,1]$, where w.l.o.g. the reward is bounded above by 1. To fix a concrete running example, consider the problem of online content recommendation: the items are webpages, and $f(S,u)$ is the *conversion* probability, i.e. the probability that a user of type $u$ will visit at least one of the webpages in $S$ if they are recommended together – we will expand on this example later in this section.

To summarize so far, if we were given a user of type $u \in \mathcal{M}$, we would seek to solve the problem $\max_{S\subset V,|S|\leq k} f(S,u)$. We will see later that this problem is 'easy' in many reasonable settings. Instead, one of the two primary challenges we seek to address in this paper is how to deal with *user heterogeneity*, i.e. when the user type is not known exactly ex-ante. We will assume that this uncertainty is modeled as a random variable $U$ over $\mathcal{M}$, whose distribution we know. Our goal then is to solve the following stochastic optimization problem:

$$\text{OPT} \equiv \max_{S\subset V,|S|\leq k} \mathsf{E}\left[f(S,U)\right]. \tag{1}$$

For the rest of this paper, we will take $U$ to be uniformly distributed over $m$ types: $u_1,\ldots,u_m \in \mathcal{M}$. There are two motivations for this: first, for $m$ sufficiently large, this assumption is without loss, as replacing the expectation in (1) with a sample average approximation yields negligible loss. Lemma 3 in the Appendix shows that $m = \Omega(k \log n)$ is sufficiently large, where $n \equiv |V|$ is the number of items. Second, in practice, what is often done is past observations of a given user are translated to points in $\mathcal{M}$, and $U$ is taken to be uniform over these points.

As was described in the Introduction, we seek to solve (1) in online settings in which the number of items $n$ is massive and the optimization must be performed extremely fast, often so fast that even algorithms linear in $n$ are infeasible. Thus, the second primary challenge we face is to solve, or approximate, problem (1) in *sub-linear* time: $o(n)$. This will require three assumptions, which we will describe in detail in the remainder of this section. Imposing these assumptions will allow us to guarantee (in expectation) an approximation of OPT using a randomized algorithm whose expected runtime, amortized over multiple users, is $O(n^{1-\epsilon})$ for some strictly positive $\epsilon$.

Our first assumption is that the reward function for any user type be monotone submodular:

ASSUMPTION 1. *For every $u \in \mathcal{M}$, the function $f(\cdot, u)$ is monotone submodular.*

The set of reward functions satisfying Assumption 1 is rich enough to include the conversion function for recommendation problems and a subclass of assortment optimization problems against the mixed multinomial logit choice model. Making this assumption is the first step in achieving sub-linearity. In fact, Assumption 1 already implies that $(1-1/e)$OPT can be guaranteed in *linear* time, as the greedy algorithm is $(1-1/e)$-optimal for maximizing monotone submodular functions subject to cardinality constraint (Nemhauser et al. (1978)). Since sums of monotone submodular functions are monotone submodular, the greedy algorithm for (1) runs in $O(kmn)$ time.

## 2.1. User and Item Embedding

The remaining two assumptions we make will allow us to use the machinery of approximate nearest neighbor algorithms in order to improve from linear to sub-linear time. To discuss these, we will first need to describe the underlying geometry of our problem. Recall that we model user types as elements of a set $\mathcal{M}$, which so far is an arbitrary set. We will assume that $\mathcal{M}$ is in fact a metric space, equipped with a metric denoted by $d(\cdot, \cdot)$. We will also assume that our universe of items is embedded in this same space: $V = \{v_1, \ldots, v_n\} \subset \mathcal{M}$.

Such embeddings are ubiquitous in predictive algorithms for personalization which, by and large, operate by estimating feature representations of users and items so that, loosely speaking, a user will have a stronger preference for items whose features 'align' more closely to his or her own features (or equivalently, those items whose features are closer in distance with respect to a carefully calibrated metric). The metric space will often either be Euclidean space or the unit-ball in Euclidean space, but using the Euclidean metric is not a requirement. Instead, what we will need to assume about our space is that there exists an appropriate data structure that returns approximate near neighbors in sub-linear time:

Assumption 2. *For any distance $\gamma > 0$, and constants $c > 1$, $\beta \in [0, 1)$, and $\epsilon \in (0, 1]$, there exists a (randomized) data structure $\mathrm{ANN}[V, \gamma, c, \beta, \epsilon] : \mathcal{M} \to 2^V$, and a corresponding $\alpha \equiv \alpha(c, \beta, \epsilon) < 1$ such that, given any query point $u \in \mathcal{M}$:*

1. *If*

$$\sum_{v \in V} \mathbb{1}(d(v, u) \le c\gamma) \le n^\beta,$$

*then for each $v \in V$ such that $d(v, u) \le \gamma$,*

$$\mathsf{P}\left(v \in \mathrm{ANN}[V, \gamma, c, \beta, \epsilon](u)\right) \ge 1 - \epsilon.$$

2. *The runtime of querying this data structure is $O(n^\alpha)$.*

*Here, constants suppressed by the big-Oh notation depend only on $\mathcal{M}$ (e.g. dimensionality).*

Assumption 2 is stated in the form typically taken in theoretical guarantees for approximate near neighbor algorithms. In words, the data structure assumed here takes any point in $\mathcal{M}$ as input, and outputs every item in $V$ that is within a pre-specified distance of the input, each with constant probability. Most importantly, the runtime of this operation is sub-linear, assuming that the number of these near neighbors is sub-linear. If $\epsilon$ could be taken to be 0, and $c$ could be taken to be 1, this would correspond to an exact near neighbor algorithm. Having $\epsilon > 0$ reflects the fact that near neighbors are not guaranteed to be returned, and having $c > 1$ reflects that in the process, elements of $V$ slightly further than $\gamma$ are generated as candidates and need to be pruned. Data structures satisfying Assumption 2 are known for a variety of metric spaces, including Euclidean space. As we will review later on, one way to construct such a structure is to use a family of hash functions with certain 'nice' properties. Our ability to solve (1) in sub-linear time will rely on our ability to sample from a certain distribution on $\mathcal{M}$. The algorithm we develop will rely on a carefully constructed ensemble of data structures of the type defined by Assumption 2

Finally, Assumptions 1 and 2 deal with the reward function $f$ and the underlying metric space $(\mathcal{M}, d)$ separately, but so far there is nothing rigorously tying the two together which would allow us to leverage the metric structure. This is the purpose of our final assumption, which states that the distance between the embeddings of a user and an item directly encodes the corresponding reward for offering that item alone to that user:

Assumption 3. *There exists a non-increasing function* $p : \mathbb{R}^+ \to [0,1]$ *such that*

$$p(d(v,u)) \geq f(\{v\}, u) \text{ for each } u \in \mathcal{M},\ v \in V.$$

*In addition, there exists some* $\beta \in [0,1)$ *and* $c > 1$ *such that*

$$\sum_{v \in V} p\left(\frac{d(v,u)}{c}\right) \leq n^{\beta} \text{ for each } u \in \mathcal{M}.$$

The function $p(\cdot)$ captures the decreasing relation between distance in $\mathcal{M}$ and reward, and will play a crucial role in our algorithm. In particular, we will treat $p(\cdot)$ as a probability in a sampling-based approach. The second part of Assumption 3, which will allow us to make use of the approximate near neighbor in sub-linear time, assumes that for all user types $u$, the total reward gotten by offering each item individually is sub-linear. This may, for example, reflect the fact that users' appetites for content are not limited by a lack of items, but rather a limit in time, attention, etc. Additionally, this condition is required to be robust in the following sense: there exists some $c > 1$ such that the condition above still holds if each $v \in V$ is replaced by a contracted vector $\tilde{v}$ such that $\|\tilde{v} - u\| = \|v - u\|/c$.

### 2.2. Examples

We conclude this section by describing two common models which fit the framework we have outlined and satisfy the three assumptions.

*Conversion Under Random Utility Choice Models* As described previously, the goal in recommendation problems is typically to induce *conversion*, i.e. selecting at least one of the items in the offer set (eg. clicking on one of a set of web links, or listening to one of a list of songs). In this setting, $f(S,u)$ is a *conversion function* which, for any set of items $S \subset \mathcal{V}$, is the probability of conversion when customer $u$ is offered set $S$.

Random utility models are commonly used to describe user choice behavior. One generic way of employing these models in the conversion problem is to assume that $f(S,u)$ takes the form

$$f(S,u) = \mathsf{P}\left(\max_{v \in S}(\mu(d(v,u)) + \epsilon_v) > 0\right), \tag{2}$$

where $\mu : \mathbb{R}^+ \to \mathbb{R}^+$ is a non-increasing function, and the $\epsilon_v$'s are i.i.d. mean-zero random variables. Here, $\mu(d(v,u)) + \epsilon_v$ is the random utility associated with selecting item $v$, with $\mu(\cdot)$ translating distance to a mean utility, and $\epsilon_v$ capturing idiosyncratic noise. We assume w.l.o.g. that the utility of selecting no item is identically zero. Users then choose the option (either one of the recommended items, or no item) that maximizes their utility, and (2) is the probability that the utility of any recommended item is higher than the utility of selecting nothing.

The formulation in (2) satisfies Assumption 1 immediately, and the function $p(\cdot)$ required by Assumption 3 can be constructed from the CDF of the $\epsilon_v$'s. This setup is extremely general and encodes many popular choice models. For example, taking the $\epsilon_v$'s to be Gumbel random variables yields the multinomial logit model, and allowing for random $U$ yields the mixed multinomial logit.

One commonly used choice of metric space and utility function in the conversion problem are the following: let $\mathcal{M} = \mathbb{S}^{d-1}$, i.e. the unit ball in $d$-dimensional Euclidean space. As stated earlier, this space satisfies Assumption 2. Now since we are dealing with unit-vectors, we have that $d(v,u)^2 = 2(1 - v^\top u)$, so taking $\mu(x) = 1 - x^2/2$ yields a natural form for the mean utility:

$$\mu(d(v,u)) = v^\top u.$$

Finally, it is worth noting that this formulation is compatible with a number of approaches to constructing metric space representations of users and products, ranging from simple logistic regression, to collaborative filtering, to state of the art approaches such as factorization machines (Rendle

(2010)) and field-aware factorization machines (Juan et al. (2016)), the latter having been a key component in the winning entries of three major recent public prediction competitions.

*Assortment Optimization Under the Mixed Multinomial Logit Model* In the field of Operations Management, a classic problem is to select an assortment of products to offer to customers so as to maximize expected revenue. Let $r_j$ be the revenue gained if a customer purchases product $v_j$. Here, we will just work out the setting where the underlying choice model is the multinomial logit:

$$f(S,u) = \sum_{v_j \in S} r_j \frac{\exp(v_j^\top u)}{w + \sum_{v \in S} \exp(v^\top u)}, \tag{3}$$

where $w \geq 0$ is a parameter that controls the likelihood that no product is selected. The objective in (3) is not in general monotone or submodular, but there are a variety of conditions which imply both. For example, one such condition shown in Han et al. (2019) is if the minimum and maximum revenues (denoted $r_{\min}$ and $r_{\max}$) are not too far apart:

$$\frac{r_{\min}}{r_{\max}} \geq \max_{S \subset V, |S| \leq k} \sum_{v_j \in S} \frac{\exp(v_j^\top u)}{w + \sum_{v \in S} \exp(v^\top u)}.$$

The expression on the right-hand side is equal to the maximum conversion (as defined previously) probability for a user of type $u$ across all feasible assortments. In particular, the revenues are allowed to vary more when this quantity is small, or equivalently, when $w$ is large. The required upper bound on $f(\{v\}, u)$ can be gotten by treating all revenues as $r_{\max}$.

## 3. Algorithm Overview

Before describing our approach, we could first consider whether brute force pre-computation would suffice, that is, simply solving (1) in advance for a comprehensive set of different distributions $U$. If feasible, this would certainly qualify as an amortized sub-linear (constant, in fact) time algorithm. There are at least two reasons why this approach might be infeasible or at best impractical. First, the size of a ‘comprehensive’ set of distributions $U$ could be massive – even in our case where $U$ is uniformly distributed over $m$ points of $V$, this set is $O(n^m)$ – in which case it may be impossible to compute and/or store it. Second, in almost all settings, the product set is dynamic. For example, the universe of online content is constantly changing. Thus, the data structure needs to be dynamic, ideally capable of fast additions and deletions. The structure we describe in the next section will allow these updates in sub-linear time.

At a high level, our algorithm proceeds in two steps:

(a) Randomly sample a sub-linear sized subset of $V$, which we will denote by $\tilde{V}$, such that if (1) is solved over $\tilde{V}$ instead of $V$, we are still guaranteed a constant fraction $(1-\epsilon)$ of OPT in expectation.

(b) Approximately solve (1) over $\tilde{V}$ using the greedy algorithm.

The crux of our algorithm is the ability to perform step (a) in sub-linear time. Assuming that step (a) could be performed in sub-linear time, the greedy algorithm in step (b) would then also run in sub-linear time, and the algorithm as a whole would be guaranteed $(1-1/e)(1-\epsilon)$ of OPT in expectation.

Ignoring the runtime of step (a) for a moment, we will first describe an idealized random sampling scheme over the items of $V$ that would return a random subset $\tilde{V}$ that is both sub-linear in size and guaranteed (in expectation) to preserve a constant fraction of OPT (less a small additive error) when optimized over. To ease notation, we will denote the objective function of (1) by $g(S) = \mathsf{E}[f(S,U)]$.

Suppose that we could sample $\tilde{V}$ such that

$$\mathsf{P}(v \in \tilde{V}) = g(\{v\}) \text{ for each } v \in V, \tag{4}$$

that is, the likelihood of any item being included in $\tilde{V}$ is equal to the proportion of OPT that the item would yield when offered alone. The following Lemma shows that making sufficiently many independent draws from such a sampling distribution, and taking the union of these draws, would result in a subset of $V$ that is guaranteed a constant fraction of OPT if subsequently optimized over:

Lemma 1. *For $c \in (0,1]$, let $\tilde{V}$ be a random variable taking values in $2^V$ such that*

$$\mathsf{P}(v \in \tilde{V}) \geq cg(\{v\}) \ \textit{for each } v \in V,$$

*and for $s \in \mathbb{N}$, let $\tilde{V}_s$ denote the union of $s$ sets drawn i.i.d. from this distribution.*

*Let $S^*(\tilde{V}_s)$ be an optimal solution to:*

$$\max_{S \subset \tilde{V}_s, |S| \le k} g(S).$$

*Then for any $\epsilon_1, \epsilon_2 \in (0,1]$, if*

$$s \ge \frac{k}{c\epsilon_2} \log \frac{k}{\epsilon_1},$$

*then we have*

$$\mathsf{E}[g(S^*(\tilde{V}_s))] \ge (1-\epsilon_1)\mathrm{OPT} - \epsilon_2.$$

*Proof of Lemma 1* Fix any $\delta \in [0,1]$ (we will tune this quantity in the end). For any $v \in V$ such that $g(\{v\}) \ge \delta$, we have

$$\begin{aligned} \mathsf{P}(v \notin \tilde{V}_s) &= \mathsf{P}(v \notin \tilde{V})^s \\ &\le (1 - cg(\{v\}))^s \\ &\le (1 - c\delta)^s \\ &\le e^{-sc\delta}, \end{aligned} \tag{5}$$

where the first equality follows from the definition of $\tilde{V}_s$, and the first two inequalities are by assumption.

Now we fix any optimal solution $S^*$ to the full problem (1), and divide it into two disjoint sets:

$$S_1 = \{v \in S^* : g(\{v\}) \ge \delta\} \text{ and } S_2 = \{v \in S^* : g(\{v\}) < \delta\}.$$

Then we have

$$\begin{aligned} g(S^*(\tilde{V})) &\ge g(S_1 \cap \tilde{V}) \\ &\ge \mathsf{P}(v \in \tilde{V} \ \ \forall\, v \in S_1) g(S_1) \\ &\ge \left(1 - \sum_{v \in S_1} \mathsf{P}(v \notin \tilde{V})\right) g(S_1) \\ &\ge \left(1 - ke^{-sc\delta}\right) g(S_1) \\ &\ge \left(1 - ke^{-sc\delta}\right) (g(S^*) - g(S_2)) \\ &= \mathrm{OPT} - ke^{-sc\delta}\mathrm{OPT} - \left(1 - ke^{-sc\delta}\right) g(S_2), \end{aligned} \tag{6}$$

where the first line is due to the optimality of $S^*(\tilde{V})$ among all solutions contained in $\tilde{V}$, the third line is a union bound, the fourth line is due to (5), and the fifth line is due to submodularity.

To conclude, it will suffice to upper bound the second and third terms in (6) by $\epsilon_1$OPT and $\epsilon_2$, respectively. To do this, we choose $\delta = \epsilon_2/k$. For the second term, applying this choice of $\delta$, along with our condition on $s$, yields the following bound:

$$ke^{-sc\delta}\text{OPT} \leq ke^{-\log(k/\epsilon_1)}\text{OPT} = \epsilon_1\text{OPT}.$$

For the third term, by submodularity and the definition of $S_2$,

$$\left(1 - ke^{-sc\delta}\right) g(S_2) \leq g(S_2) \leq \sum_{v \in S_2} g(v) \leq k\delta = \epsilon_2.$$

Lemma 1 shows that to approximate OPT to arbitrary precision in expectation, it suffices to sample $s = O(k \log k)$ times from a distribution approximately satisfying (4). In fact, the Lemma states that the sampling probabilities do not need to match (4), but that they just need to be lower bounded by some constant fraction $c$ of (4). In the algorithm we outline later, we will arbitrarily take this fraction to be $c = 1/2$.

Having guaranteed that a constant fraction of OPT is preserved, the other required condition on (4) is that the resulting subset be sub-linear in size, as the greedy algorithm that follows is linear in the size of this set. The expected size of $\tilde{V}$ sampled according to (4) is

$$\mathsf{E}[|\tilde{V}|] = \sum_{v \in V} g(\{v\}) \leq n^\beta, \tag{7}$$

which is guaranteed to be sub-linear by Assumption 3.

## 4. Our Approach in Detail: Locality-Sensitive Sampling

### 4.1. Approximating the Ideal Sampling Distribution via Locality-Sensitive Sampling

Now with the goal of executing the ideal sampling distribution (4), the brute-force method to sample exactly from this distribution would be to generate $n$ independent Bernoulli variables, whose means are $g(\{v\})$ for each $v \in V$. By Lemma 1, repeating this procedure $s = \Omega(k \log k)$ times yields a pruned set of items that preserves a good approximation of OPT in expectation. However, generating the $mn$ Bernoulli variables is clearly a linear time procedure. Fortunately, it is possible in sub-linear time to *approximate* (4), i.e. the probabilities in (4) are not matched exactly, but rather just up to a constant:

$$\mathsf{P}(v \in \tilde{V}) \sim g(\{v\}) \text{ for each } v \in V.$$

The key observation that makes this possible is that while the probabilities of the individual events $\{v_j \in \tilde{V}\}$ need to be strictly controlled, these events are allowed to be arbitrarily correlated. This is precisely what allows us to leverage the underlying metric space, along with the approximate near neighbor data structures assumed by Assumption 2, to approximately perform (4).

To see why this is possible, first note that allowing arbitrary correlations implies that to sample each $v$ with probability $g(\{v\}) = \mathsf{E}[f(\{v\}, U)]$, it suffices to first draw $u$ according to $U$, and then sample each $v$ with probability $f(\{v\}, u)$. Next recall that by Assumption 3, there exists a function $p(\cdot)$ such that $f(\{v\}, u) \le p(d(v, u))$. This allows us to use near neighbor queries to do the sampling As a simple example, if $p(x) = \mathbb{1}(x \le \gamma)$ for some $\gamma$, then sampling with probability $f(\{v\}, u)$ is equivalent to returning all $v \in V$ within distance $\gamma$ of $u$, and thus a single approximate near neighbor data structure suffices.

More generally, our algorithm utilizes $R = \lfloor \log_2 n^{1-\beta} \rfloor$ of these structures to approximate any non-increasing function $p(\cdot)$. For each $r \in [R]$, let

$$\rho_r = \begin{cases} 1/(2^r - 1), & r \in [R-1] \\ 1/2^{r-1}, & r = R \end{cases} \quad \text{and} \quad \gamma_r = \sup\{x : p(x) \ge 1/2^r\}. \tag{8}$$

See Figure 1 for a visual depiction of these parameters and our strategy, which is to approximate $p(\cdot)$ by a step function, each step represented by a single near neighbor data structure.

Our overall scheme then is to create a set of approximate near neighbor structures, and sample from $p(d(v, u))$ by querying each structure and returning their union. The various parameters for these structures are given by the $\rho_r$'s and $\gamma_r$'s, along with our choice of $\epsilon = 1/2$ (chosen arbitrarily to save on notation).

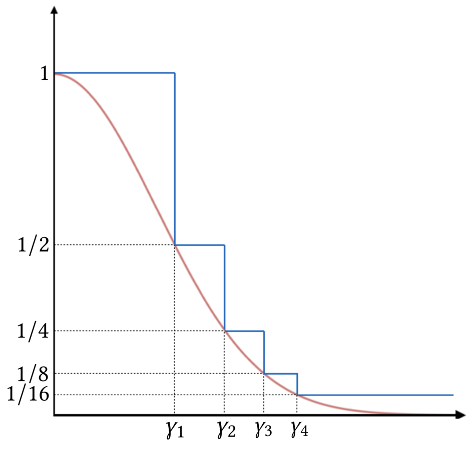

**Figure 1** **Visual depiction of the approximate sampling scheme. The red curve contains the ideal sampling probabilities $p(\cdot)$, and the blue curve shows how we attempt to approximate it using a step function. In this example, $R = 4$.**

Definition 1 (Locality-Sensitive Sampling). Let $V$ be a finite subset of a metric space $\mathcal{M}$ satisfying Assumption 2, For any non-increasing function $p : \mathbb{R}^+ \to [0,1]$, and any constant $c > 1$, the *Locality-Sensitive Sampling* data structure is a (randomized) map $\mathrm{LSS}[V,p,c,\beta] : \mathcal{M} \to 2^V$:

$$\mathrm{LSS}[V,p,c,\beta](u) = \rho_0 V \bigcup \left( \bigcup_{r=1}^{R} \mathrm{ANN}[\rho_r V, \gamma_r, c, \beta, 1/2](u) \right) \quad \text{for all } u \in \mathcal{M},$$

where the $\rho_r$ and $\gamma_r$ are defined as in (8), $\rho_0 = 2n^{\beta-1}$, and each $\rho_r V$ denotes a random subset of $V$ gotten by including each element of $V$ independently with probability $\rho_r$.

Lemma 2. *For each $u \in \mathcal{M}$ and $v \in V$,*

$$\mathsf{P}(v \in \mathrm{LSS}[V,p,c,\beta](u)) \geq p(d(v,u))/2.$$

*Moreover, each query $\mathrm{LSS}[V,p,c,\beta](u)$ has runtime*

$$O\left(n^{\alpha} \log n\right),$$

*where $\alpha = \alpha(c,\beta,1/2)$.*

## 4.2. Aside: LSS Using Locality-Sensitive Hash Functions

So far, we have assumed the existence of ANN data structures satisfying Assumption 2, without describing how any of these work. In this subsection, we describe one existing approach based on locality-sensitive hash (LSH) functions (originally described in the seminal work of Indyk and Motwani (1998)), and show how an LSS structure can be constructed from scratch from these functions. This subsection can be safely skipped without loss of continuity.

The key component of LSH algorithms are *LSH families*: let $\mathcal{H}$ be a family of functions defined on $\mathcal{M}$ such that when $h$ is chosen uniformly at random from $\mathcal{H}$, we have $\mathsf{P}(h(u_1) = h(u_2)) = q(d(u_1, u_2))$. Here, $q : [0, \infty) \to [0, 1]$ is some non-increasing function such that $q(0) = 1$ and $q(x) > 0$ if $p(x) > 0$. We will show that sampling from $V$ can be approximated in sublinear time using an LSH family $\mathcal{H}$ and our Locality-Sensitive Sampling procedure:

Proposition 1. *Let $\mathcal{H}$ and $q$ be defined as in the preceding text, and suppose that*

$$\log_{q(cx)} q(x) \leq \delta \quad \textit{for all } x \textit{ and some } \delta < 1.^{2}$$

*Then there exists a locality-sensitive sampling data structure built from these hash functions such that Lemma 2 holds with*

$$\alpha = \beta + \delta(1 - \beta).$$

Proposition 1 is only useful assuming the existence of a family of functions $\mathcal{H}$ satisfying the condition in the statement of the Proposition. Does such a family in fact exist? The search and analysis of appropriate families of functions for various metric spaces has been an active area of research. For our own setting, where $\mathcal{M} = \mathbb{S}^{d-1}$ and the metric is induced by the $\ell_2$ norm, there are recent results (Terasawa and Tanaka (2007), Andoni and Razenshteyn (2015), Andoni et al. (2015)) for the *cross-polytope* hash family that essentially amounts to randomly rotating a set of pre-defined points on the sphere and hashing each vector to its nearest point. Even simpler is the *hyperplane* LSH family where each function corresponds to a single vector, and the function assigns to any vector the sign of its inner product with the defining vector. Charikar (2002) show that $\delta$ can be taken to be $1/c$ using this family.

[2] We follow the convention that $\log_0 x = 0$ for any $x \in [0, 1]$.

To describe the locality-sensitive sampling procedure, we begin by defining the vanilla LSH data structure, which we parameterize by $\rho \in [0,1]$ and integers $a, b > 0$. To construct the data structure, first a random subset $\rho V \subset V$ is taken by including each element of $V$ independently with probability $\rho$. Then a total of $b$ hash tables are constructed, with each table storing all of the items in $\rho V$. The hash function for each table $j = 1, \ldots, b$ is vector-valued, constructed by drawing functions $h_1^j, \ldots, h_a^j$ independently and uniformly at random from $\mathcal{H}$. This entire construction is done during the preprocessing phase. Then given a query point $u \in \mathcal{M}$, we hash $u$ in each table and return all collisions:

$$\mathrm{LSH}_{\rho,a,b}(u) = \left\{ v \in \rho V : (h_1^j(v), \ldots, h_a^j(v)) = (h_1^j(u), \ldots, h_a^j(u)) \text{ for some } j \in [b] \right\}.$$

Thus, $\mathrm{LSH}_{\rho,a,b}(u)$ is a random subset of $V$, where the randomness is with respect to the sampling when creating $\rho V$ and selecting the hash functions.

Our locality-sensitive sampling algorithm then utilizes $R = \lceil 1 + \log_2 n \rceil$ LSH structures. For each $r \in [R]$, let $\rho_r$ and $\gamma_r$ be defined as in (8). Moreover, let

$$a_r = \left\lceil \log_{q(c\gamma_r)} 2^r n^{\beta-1} \right\rceil \text{ and } b_r = \left\lceil \log(2) 2^{-r\delta} n^{\delta(1-\beta)} (1/q(\gamma_r)) \right\rceil.$$

Then given a query $u$, for each $r \in [R]$, we calculate $\mathrm{LSH}_{\rho_r,a_r,b_r}(u)$ and return their union:

$$\bigcup_{r=1}^{R} \mathrm{LSH}_{\rho_r,a_r,b_r}(u).$$

### 4.3. Putting It All Together

Through locality-sensitive sampling, we now have a method of approximating our ideal sampling scheme (4). Lemma 1 requires that $\tilde{V}$ be constructed from $s = \Omega(k \log k)$ *independent* samples from this distribution, so we require $s$ instances of this overall structure (each LSS structure itself a combination of ANN structures).

Our final step then is to solve

$$\max_{S \subset \tilde{V}, |S| \le k} g(S)$$

using the greedy algorithm, where recall that $g(S) = \mathsf{E}[f(S,U)]$. Specifically, this problem is one of maximizing a monotone submodular set function under cardinality constraint, and as such, is known to admit a $1-e^{-1}$ approximation via a greedy algorithm (Nemhauser et al. (1978)). Note that the $1-e^{-1}$ guarantee is the best-known guarantee among polynomial-time algorithms; indeed, even the conversion problem under the no-noise case ($\epsilon = 0$) falls into a class of geometric set cover problems known to be APX-hard (Mustafa et al. (2014)).

The greedy algorithm constructs a solution sequentially as follows: at step $\ell$, having already constructed set $S_{\ell-1}$, we choose $S_\ell$ to be

$$S_\ell = S_{\ell-1} \cup \arg\max_{v \in \tilde{V}} g(S_{\ell-1} \cup \{v\}),$$

where ties are broken arbitrarily. Initiating $S_0$ to be the empty set, the algorithm completes in $k$ steps. Each step of this greedy algorithm requires calculating $g(S_{\ell-1} \cup \{v\})$ for each $v$ in $\tilde{V}$, with each evaluation taking $O(m)$ time. Therefore, the entire greedy procedure runs in $O(km|\tilde{V}|)$ time. To summarize, the last two sections have shown that our algorithm successfully achieves an approximation in sub-linear time.

Theorem 1. *For any $\epsilon_1, \epsilon_2 \in (0,1]$, there exists a data structure and algorithm that achieves*

$$(1-e^{-1})[(1-\epsilon_1)\mathrm{OPT} - \epsilon_2]$$

*in expectation and has amortized runtime*

$$O\left((n^\alpha \log n + kmn^\beta)\frac{k}{\epsilon_2}\log\frac{k}{\epsilon_1}\right),$$

*where $\alpha = \alpha(c, \beta, 1/2)$.*

## 5. Experiments

We performed two sets of experiments to illustrate different aspects of our modeling approach and the LSS algorithm.

## 5.1. Locality Sensitive Sampling

In the first set of experiments, we demonstrate the sampling achieved by the locality-sensitive sampling scheme on the target function defined by the multinomial logit choice probabilities. We synthetically generated the set of vectors $V$ and the user vector $u$ in $\mathbb{S}^{49}$. $u$ is fixed as the unit vector $(-1, 0, \cdots 0)$. $V$ is generated such that the distance between $u$ and the vectors in $V$ follows a uniform distribution with support [0, 2]. This guarantees that there are adequate number of points in $V$ for all possible distance values from the user vector. The function $p(\cdot)$ is defined based on the probability of click in a MNL choice model:

$$p(x) = \begin{cases} 1 - \frac{v_0}{v_0 + e^{1-\frac{x^2}{2}}} & 0 \le x < \theta \\ 0 & \text{else.} \end{cases}$$

where the parameter $\theta$ is a threshold value such that the probability of clicking on a product is 0 if the distance between the user and the product vector is greater than $\theta$. $v_0$ captures the user's utility from not making any purchase. The sampling scheme aims to sample a product at a distance $x$ from $u$ with probability in between $0.95p(x)$ and $2p(x)$. We follow the locality sensitive sampling procedure to obtain the samples. The hash functions are based on the Hyperplane LSH family. Thus, the hash function can be characterized as $q(x) = 1 - \frac{1}{\pi} \cos^{-1}\left(1 - \frac{x^2}{2}\right)$. Python package FALCONN Andoni et al. (2015) is used to build the LSH strcutures. A family of these LSH structures is built with appropriate values of number of tables and number of hash functions in each table as per the locality sensitive sampling procedure. Then, we query all these LSH structures with the user vector and obtain all the returned points as samples.

We estimate the sampling probability of a product at a distance $x$ by repeating the locality sensitive sampling procedure 20 times on the dataset and then measuring the fraction of times that the products at distance $x$ were sampled. (The hash functions are chosen randomly in each of these repetitions leading to different LSH structures.) Values of all the parameters used in this experiment are summarized in Table 2.

The sampling probabilities, along with $p(x)$ and the upper and lower bounds of $2p(x)$ and $0.95p(x)$ are plotted in Figure 3. We observe that, the locality sensitive sampling scheme effectively samples products as per the desired distribution.

| **Parameter** | **Value** |
|---|---|
| Number of products ($n$) | 50000 |
| Dimension ($d$) | 50 |
| No purchase parameter ($v_0$) | 10 |
| Lower bound parameter ($\epsilon$) | 0.05 |
| Threshold ($\theta$) | $\sqrt{2}$ |
| LSH parameter ($c$) | 1.1 |
| LSH parameter ($\beta$) | 0.3 |

**Figure 2** **Values of parameters taken in the synthetic data experiment.**

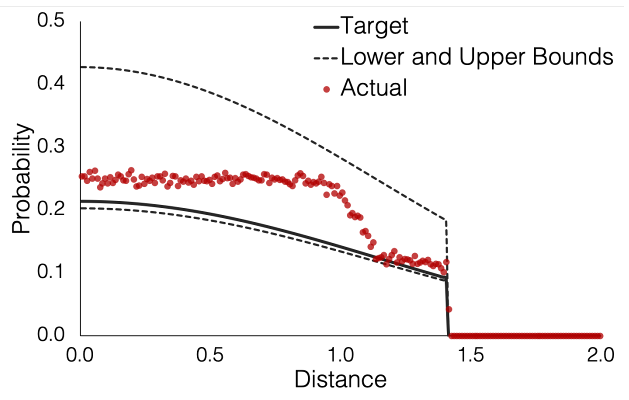

**Figure 3** **Sampling of products using Locality Sensitive Sampling. The sampling probability achieved by the locality sensitive sampling procedure along with the upper and lower bounds of the target distribution are shown.**

## 5.2. Modeling Diversity in User Behavior

In the second experiment, using a real data set from Outbrain, we illustrate the value that can be gained from using a finer user representation in place of the current practice of representing the user as a fixed vector. Outbrain is a content delivery platform providing content recommendation on websites of numerous publishers .Outbrain serves over 250 billion personalized content recommendations every month and reaches over 565 million unique visitors Outbrain (2017). Their promoted articles appear on more than 35,000 websites, reaching over 87% of US Internet users Outbrain (2017).

Our dataset contains a sample of pages viewed and clicked by users on multiple publisher sites in the United States between 14 June, 2016 and 28 June, 2016. The dataset contains two billion page views of 700 million unique users across 560 websites, amounting to around 100 GB of data.

We considered documents which have been viewed more than 750 times and users who have viewed more than 30 pages. There are approximately $10^6$ such documents and $10^7$ such users.

The users were split into test (containing 1000 users) and training sets (containing the remaining users). We then constructed a representation of each web page in a latent feature space using the *prod2vec* model Grbovic et al. (2015). For this, we made a chronological list of pages viewed by each user in the training set, and then concatenated these lists. This was then interpreted as a list of 'sentences' from which we learnt a representation of each 'word' i.e. web page using *word2vec* Mikolov et al. (2013a,b). We briefly describe the *word2vec* method.

*word2vec* is a state of the art method to construct representation of words in a vector space. It consists of two types of models - Continuous Bag of Words (CBOW) and Skip Gram model. The CBOW models relies on the bag of words assumption. It classifies a page based on its context i.e. its surrounding pages, disregarding the order of the surrounding pages. The Skip Gram Model uses the order of the context pages - giving more weight to nearby pages in its context than pages which are farther away. This is a two layer neural network which is trained to predict probabilities of pages which are nearby for a given page. After the training, the weight matrix of the hidden layer of the neural network gives the representation of the words in Euclidean space.

Owing to its superior representation of infrequent words, the skip gram model in *word2vec* was used to construct an embedding of the pages in $\mathcal{R}^{100}$ using this corpus of page views.[3] We normalized these vectors to lie on the unit sphere. For finding the representation of a user in the feature space, different user models were used for different algorithms. We will describe these along with the description of the implemented algorithms.

We consider the task of making 500 page recommendations for a sample of 1000 test users based on their history of 10 page views. We make recommendation as per LSS and also based on two commonly used algorithms which serve as benchmarks for evaluating the performance of LSS.

In LSS, the historical page views of the user are taken as samples from the distribution of $U$. We consider the no noise case i.e. $\epsilon = 0$ and fix radius $\gamma = 1.4$ which encodes the option that

[3] We used the implementation of *word2vec* in Python Machine Learning Library (MLlib), built on Apache Spark.

| **Algorithm** | LSS | Mean | Last Viewed |
|---|---|---|---|
| **Hits (%)** | 8.5 | 7.5 | 4.9 |

**Table 1 Summary of Outbrain experiment. Average conversion in 500 recommendations is reported for the LSS-based algorithm and two current practice benchmarks.**

the user selects nothing. To make 500 recommendations, we consider a natural generalization of LSS. We make recommendations according to the greedy algorithm until all samples $u$ in the user history have a page $v$ in the set of recommendations such that $v^T u > \gamma$. Then, we set aside these recommendations and re-run the greedy algorithm. We continue doing this till we have 500 recommendations.

We also make recommendations according to two benchmark algorithms. In both these benchmarks, we find a representation of the user vector. Then, the 500 pages closest to this user vector are recommended. In the *Mean* benchmark, the user vector is the mean of all the pages in the user's history. In the *Last Viewed* benchmark, the user vector is the vector representation of the last page viewed by the user (in the history given to us). Both these algorithms are prevalent in industry to make recommendations.

To evaluate the performance of these algorithms, we count the number of hits i.e. number of pages in the recommendations that were actually viewed by the user. The results are presented in Table 1. LSS gives a 13.3% and 73.4% improvement in the number of hits as compared to *Mean* and *Last Viewed* benchmarks respectively. This is a significant improvement. Intuitively, being able to model each user as a distribution helps in capturing the diversity in the user's behaviour. The recommendations can then be made keeping in mind these different aspects of the user's behaviour.

## References

Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering*, 17(6):734–749, 2005.

Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Communications of the ACM*, 51(1):117, 2008.

Alexandr Andoni and Ilya Razenshteyn. Optimal data-dependent hashing for approximate near neighbors. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*, pages 793–801. ACM, 2015.

Alexandr Andoni, Piotr Indyk, Thijs Laarhoven, Ilya Razenshteyn, and Ludwig Schmidt. Practical and optimal lsh for angular distance. In *Advances in Neural Information Processing Systems*, pages 1225–1233, 2015.

Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

Moses S Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 380–388. ACM, 2002.

Edgar Chávez, Gonzalo Navarro, Ricardo Baeza-Yates, and José Luis Marroquín. Searching in metric spaces. *ACM computing surveys (CSUR)*, 33(3):273–321, 2001.

Abhinandan S Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. Google news personalization: scalable online collaborative filtering. In *Proceedings of the 16th international conference on World Wide Web*, pages 271–280, 2007.

James Davis, Guillermo Gallego, and Huseyin Topaloglu. Assortment planning under the multinomial logit model with totally unimodular constraint structures. *Department of IEOR, Columbia University. Available at http://www. columbia. edu/ gmg2/logit_const. pdf*, 2013.

Antoine Désir, Vineet Goyal, Danny Segev, and Chun Ye. Capacity constrained assortment optimization under the markov chain based choice model. *Operations Research, Forthcoming*, 2015.

Antoine Désir, Vineet Goyal, and Danny Segev. Assortment optimization under a random swap based distribution over permutations model. In *EC*, pages 341–342, 2016.

Vivek F Farias, Srikanth Jagabathula, and Devavrat Shah. A nonparametric approach to modeling choice with limited data. *Management Science*, 59(2):305–322, 2013.

Yoav Freund, Raj Iyer, Robert E Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. *Journal of machine learning research*, 4(Nov):933–969, 2003.

Mihajlo Grbovic, Vladan Radosavljevic, Nemanja Djuric, Narayan Bhamidipati, Jaikit Savla, Varun Bhagwan, and Doug Sharp. E-commerce in your inbox: Product recommendations at scale. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1809–1818. ACM, 2015.

Shaoning Han, Andrés Gómez, and Oleg A Prokopyev. Assortment optimization and submodularity. 2019.

Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613. ACM, 1998.

Rong Jin, Luo Si, and ChengXiang Zhai. Preference-based graphic models for collaborative filtering. In *Proceedings of the Nineteenth conference on Uncertainty in Artificial Intelligence*, pages 329–336. Morgan Kaufmann Publishers Inc., 2002.

Rong Jin, Luo Si, ChengXiang Zhai, and Jamie Callan. Collaborative filtering with decoupled models for preferences and ratings. In *Proceedings of the twelfth international conference on Information and knowledge management*, pages 309–316. ACM, 2003.

Yuchin Juan, Yong Zhuang, Wei-Sheng Chin, and Chih-Jen Lin. Field-aware factorization machines for ctr prediction. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 43–50. ACM, 2016.

Alexandros Karatzoglou, Alex Smola, and Markus Weimer. Collaborative filtering on a budget. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 389–396, 2010.

Jon Kleinberg, Sendhil Mullainathan, and Johan Ugander. Comparison-based choices. In *Proceedings of the 2017 ACM Conference on Economics and Computation*, pages 127–144, 2017.

A Gürhan Kök, Marshall L Fisher, and Ramnath Vaidyanathan. Assortment planning: Review of literature and industry practice. In *Retail supply chain management*, pages 99–153. Springer, 2008.

Matevž Kunaver and Tomaž Požrl. Diversity in recommender systems–a survey. *Knowledge-Based Systems*, 123:154–162, 2017.

Han Liu, Xiangnan He, Fuli Feng, Liqiang Nie, Rui Liu, and Hanwang Zhang. Discrete factorization machines for fast feature-based recommendation. *arXiv preprint arXiv:1805.02232*, 2018.

Xianglong Liu, Junfeng He, Cheng Deng, and Bo Lang. Collaborative hashing. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2139–2146, 2014.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013a.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013b.

Nabil H Mustafa, Rajiv Raman, and Saurabh Ray. Settling the apx-hardness status for geometric set cover. In *Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on*, pages 541–550. IEEE, 2014.

George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functionsi. *Mathematical Programming*, 14(1):265–294, 1978.

Stephen M Omohundro. *Five balltree construction algorithms.* International Computer Science Institute Berkeley, 1989.

Outbrain. Similar tech, 2017. URL `https://www.similartech.com/technologies/outbrain`. [Online; accessed April 13, 2017].

Loïc Paulevé, Hervé Jégou, and Laurent Amsaleg. Locality sensitive hashing: A comparison of hash function types and querying mechanisms. *Pattern Recognition Letters*, 31(11):1348–1358, 2010.

Steffen Rendle. Factorization machines. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 995–1000. IEEE, 2010.

Paat Rusmevichientong, Zuo-Jun Max Shen, and David B Shmoys. Dynamic assortment optimization with a multinomial logit choice model and capacity constraint. *Operations research*, 58(6):1666–1680, 2010.

William W Cohen Robert E Schapire and Yoram Singer. Learning to order things. *Advances in Neural Information Processing Systems*, 10(451):24, 1998.

Robert F Sproull. Refinements to nearest-neighbor searching in k-dimensional trees. *Algorithmica*, 6(1): 579–589, 1991.

Kalyan Talluri and Garrett Van Ryzin. Revenue management under a general discrete choice model of consumer behavior. *Management Science*, 50(1):15–33, 2004.

Kengo Terasawa and Yuzuru Tanaka. Spherical lsh for approximate nearest neighbor search on unit hypersphere. In *Workshop on Algorithms and Data Structures*, pages 27–38. Springer, 2007.

Huw CWL Williams. On the formation of travel demand models and economic evaluation measures of user benefit. *Environment and planning A*, 9(3):285–344, 1977.

Peter N Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *SODA*, volume 93, pages 311–21, 1993.

Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)*, 52(1):1–38, 2019.

Zhiwei Zhang, Qifan Wang, Lingyun Ruan, and Luo Si. Preference preserving hashing for efficient recommendation. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 183–192, 2014.

Ke Zhou and Hongyuan Zha. Learning binary codes for collaborative filtering. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 498–506, 2012.

## Appendix A: Sample Complexity of the Sample Average Approximation

Lemma 3. *Let $U_1, \dots, U_m$ be i.i.d. samples from distribution $U$, and let $S_m^*$ be an optimal solution to*

$$\max_{S \subset V, |S| \le k} \frac{1}{m} \sum_{i \in [m]} f(S, U_i).$$

*There exists some universal constant $C$ such that for any $\epsilon \in (0,1]$,*

$$\mathsf{E}[f(S_m^*, U)] \ge \mathrm{OPT} - \epsilon$$

*as long as*

$$m \ge C \frac{k \log n}{\epsilon^2}.$$

*Proof of Lemma 3* This follows from a standard tail bound for bounded (or sub-gaussian, more generally) variables. Fix any $t > 0$. For any $S \subset V$, the random variable $f(S,U)$ lies in the interval $[0,1]$ by assumption, and so by Hoeffding's inequality,

$$\mathsf{P}\left(\left|\frac{1}{m}\sum_{i\in[m]} f(S,U_i) - \mathsf{E}[f(S,U)]\right| > t\right) \le 2\exp\left(-cmt^2\right),$$

for some universal constant $c$. Applying a union bound over all cardinality-constrained $S \subset V$, we obtain:

$$\begin{aligned}\mathsf{P}\left(\max_{S\subset V,|S|\le k}\left|\frac{1}{m}\sum_{i\in[m]} f(S,U_i) - \mathsf{E}[f(S,U)]\right| > t\right) &\le \sum_{S\subset V,|S|\le k}\mathsf{P}\left(\left|\frac{1}{m}\sum_{i\in[m]} f(S,U_i) - \mathsf{E}[f(S,U)]\right| > t\right)\\ &\le 2kn^k\exp\left(-cmt^2\right),\end{aligned}$$

which implies (e.g. by integrating over $t \ge 0$) that for some constant $C$,

$$\mathsf{E}\left[\max_{S\subset V,|S|\le k}\left|\frac{1}{m}\sum_{i\in[m]} f(S,U_i) - \mathsf{E}[f(S,U)]\right|\right] \le C\sqrt{\frac{k\log n}{m}}.$$

The error incurred by optimizing over the sample average approximation is at most twice this uniform bound, which equals $\epsilon$ for $m$ as given in the statement of the theorem.

## Appendix B: Additional Proofs

### B.1. Proof of Lemma 2

First, fix any $u \in \mathcal{M}$ and $v \in V$, and let $r_0 = \lceil -\log_2 p(d(v,u))\rceil$. If $p(d(v,u)) \le 2\rho_0$, then clearly

$$\mathsf{P}(v \in \mathrm{LSS}[V,p,c](u)) \ge \rho_0 \ge p(d(v,u))/2.$$

Now, let us consider the case when $p(d(v,u)) > 2\rho_0$. Here we have

$$\begin{aligned}\mathsf{P}(v \in \mathrm{LSS}[V,p,c](u)) &\ge \mathsf{P}\left(v \in \bigcup_{r=r_0}^{R}\mathrm{ANN}[\rho_r V,\gamma_r,c,1/2](u)\right)\\ &= 1 - \prod_{r=r_0}^{R}\mathsf{P}\left(v \notin \mathrm{ANN}[\rho_r V,\gamma_r,c,1/2](u)\right)\\ &\ge 1 - \prod_{r=r_0}^{R}(1-\rho_r/2)\\ &\ge \frac{1/2}{2^{r-1}}\\ &\ge p(d(v,u))/2,\end{aligned}$$

which is precisely the first statement.

Now, by Assumption 2, we can guarantee $O(Rn^{\alpha})$ runtime as long as the expected number of items returned is $O(n^{\beta})$. This is indeed the case:

$$\begin{aligned}\sum_{r=1}^{R}\rho_r\sum_{v\in V}\mathbb{1}\{d(v_j,u)\le c\gamma_r\} &= \sum_{v\in V}\sum_{r=1}^{R}\rho_r\mathbb{1}\{d(v_j,u)\le c\gamma_r\}\\ &\le \sum_{v\in V}\max\left\{2p\left(\frac{d(v,u)}{c}\right),\frac{1}{n}\right\}\\ &\le 2n^{\beta}\end{aligned}$$

Q.E.D.

## B.2. Proof of Proposition 1

Two facts follow from the choice of parameters in the LSH data structures. First, for any $v$ such that $p(u,v)\in(\rho_r/2,\rho_r]$, the probability that the algorithm returns $v$ is at most $\rho_r$ and at least $\rho_r[1-(1-q(\gamma_r)^{a_r})^{b_r}]$. Since we have

$$\begin{aligned}(1-q(\gamma_r)^{a_r})^{b_r} &\le \exp\left(-q(\gamma_r)^{a_r}b_r\right)\\ &\le \exp\left(-q(\gamma_r)(2^r n^{\beta-1})^{\log_{q(c\gamma_r)}q(\gamma_r)}\log(2)2^{-r\delta}n^{\delta(1-\beta)}(1/q(\gamma_r))\right)\\ &\le 1/2,\end{aligned}$$

it follows that this satisfies the sampling requirement.

Second, the expected total number of collisions in a given LSH structure is at most

$$\begin{aligned}(2n^{\beta}+\rho_r n q(c\gamma_r)^{a_r})b_r &\le (2n^{\beta}+\rho_r n 2^r n^{\beta-1})b_r\\ &= 4n^{\beta}b_r\\ &\le 2^{2-r\delta}n^{\beta+\delta(1-\beta)}(1/q(\gamma_r))\log(2)+4n^{\beta}.\end{aligned}$$

The first inequality follows from the definition of $a_r$, which implies that

$$a_r \ge \log_{q(c\gamma_r)} 2^r n^{\beta-1}.$$

The equality comes from the definition of $\rho_r$ and combining terms. The second inequality follows from the definition of $b_r$, which implies that

$$b_r \le \log(2)2^{-r\delta}n^{\delta(1-\beta)}(1/q(\gamma_r))+1.$$

Thus, across all structure the expected number of collisions is $O(n^{\beta+\delta(1-\beta)}\log n)$.